

5.138 global_cardinality

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	CHARME [268]			
Constraint	global_cardinality(VARIABLES, VALUES)			
Synonyms	count, distribute, distribution, extended_global_cardinality.	gcc,	card_var_gcc,	egcc,
Arguments	VARIABLES : collection(var-dvar) VALUES : collection(val-int, noccurrence-dvar)			
Restrictions	required(VARIABLES, var) required(VALUES, [val, noccurrence]) distinct(VALUES, val) VALUES.noccurrence \geq 0 VALUES.noccurrence \leq VARIABLES			
Purpose	Each value VALUES[i].val ($1 \leq i \leq \text{VALUES} $) should be taken by exactly VALUES[i].noccurrence variables of the VARIABLES collection.			
Example	$\left(\langle 3, 3, 8, 6 \rangle, \left\langle \begin{array}{l} \text{val} - 3 \quad \text{noccurrence} - 2, \\ \text{val} - 5 \quad \text{noccurrence} - 0, \\ \text{val} - 6 \quad \text{noccurrence} - 1 \end{array} \right\rangle \right)$			
	The global_cardinality constraint holds since values 3, 5 and 6 respectively occur 2, 0 and 1 times within the collection $\langle 3, 3, 8, 6 \rangle$ and since no constraint was specified for value 8.			
Typical	VARIABLES > 1 range(VARIABLES.var) > 1 VALUES > 1 range(VALUES.noccurrence) > 1 VARIABLES > VALUES			
Symmetries	<ul style="list-style-type: none"> • Items of VARIABLES are permutable. • Items of VALUES are permutable. • An occurrence of a value of VARIABLES.var that does not belong to VALUES.val can be replaced by any other value that also does not belong to VALUES.val. • All occurrences of two distinct values in VARIABLES.var or VALUES.val can be swapped; all occurrences of a value in VARIABLES.var or VALUES.val can be renamed to any unused value. 			

Usage

We show how to use the `global_cardinality` constraint in order to model the [magic series](#) problem [377, page 155] with one single `global_cardinality` constraint. A non-empty finite series $S = (s_0, s_1, \dots, s_n)$ is *magic* if and only if there are s_i occurrences of i in S for each integer i ranging from 0 to n . This leads to the following model:

$$\text{global_cardinality} \left(\left\langle \begin{array}{l} \langle \text{var} - s_0, \text{var} - s_1, \dots, \text{var} - s_n \rangle, \\ \text{val} - 0 \quad \text{noccurrence} - s_0, \\ \text{val} - 1 \quad \text{noccurrence} - s_1, \\ \quad \quad \quad \vdots \\ \text{val} - n \quad \text{noccurrence} - s_n \end{array} \right\rangle \right)$$

Remark

This is a generalised form of the original `global_cardinality` constraint: in the original `global_cardinality` constraint [309], one specifies for each value its minimum and maximum number of occurrences (i.e., see [global_cardinality_low_up](#)). Here we give for each value v a domain variable that indicates how many time value v is effectively used. By setting the minimum and maximum values of this variable to the appropriate constants we can express the same thing as in the original `global_cardinality` constraint. However, as shown in the *magic series* problem, we can also use this variable in other constraints. By reduction from [3-SAT](#), Claude-Guy Quimper shows in [298] that it is NP-hard to achieve [arc-consistency](#) for the count variables.

A last difference with the original `global_cardinality` constraint comes from the fact that there is no constraint on the values that are not explicitly mentioned in the `VALUES` collection. In the original `global_cardinality` these values could not be assigned to the variables of the `VARIABLES` collection. However allowing values that are not mentioned in `VALUES` to be assigned to variables of `VARIABLES` can potentially avoid mentioning a huge number of unconstrained values in the `VALUES` collection, and as a side effect, prevent eventually⁵ generating a dense graph (i.e., see [DFS-bottleneck](#)) for the corresponding underlying [flow model](#)).

Within [75] the `global_cardinality` constraint is called `distribution`. Within [317] the `global_cardinality` constraint is called `card_var_gcc`. Within [64] the `global_cardinality` constraint is called `egcc` or `rgcc`. This later case corresponds to the fact that some variables are duplicated within the `VARIABLES` collection.

The `global_cardinality` constraint can be seen as a system (i.e., a conjunction) of [among](#) constraints.

When all count variables (i.e., the variables `VALUES[i].noccurrence` with $i \in [1, |\text{VALUES}|]$) do not occur in any other constraints of the problem, it may be operationally more efficient to replace the `global_cardinality` constraint by a [global_cardinality_low_up](#) constraint where each count variable `VALUES[i].noccurrence` is replaced by the corresponding interval `[VALUES[i].noccurrence, VALUES[i].noccurrence]`. This stands for two reasons:

- First, by using a [global_cardinality_low_up](#) constraint rather than a `global_cardinality` constraint, we avoid the filtering algorithm related to the count variables.

⁵ Of course one could also, while generating a flow model, detect all unconstrained values in order to generate one single vertex in the flow model for the set of unconstrained values.

- Second, unlike the `global_cardinality` constraint where we need to fix all its variables to get `entailment`, the `global_cardinality_low_up` constraint can be `entailed` before all its variables get fixed. As a result, this potentially avoid unnecessary calls to its filtering algorithm.

An implicit necessary condition inferred by double counting with the `global_cardinality` constraint is depicted by the following expression:

$$\sum_{i=1}^{|\text{VARIABLES}|} \text{VARIABLES}[i].\text{var} = \sum_{i=1}^{|\text{VALUES}|} \text{VALUES}[i].\text{nocurrence} \cdot \text{VALUES}[i].\text{val}$$

Within [284, pages 50–51] the previous condition where terms involving identical variables are grouped together (i.e., rule 5 of MALICE [283]) is mentioned as a crucial deduction rule for the `autoref` problem.

W.-J. van Hove *et al.* present two soft versions of the `global_cardinality` constraint in [385].

Algorithm

A `flow` algorithm that handles the original `global_cardinality` constraint is described in [309]. The two approaches that were used to design `bound-consistency` algorithms for `alldifferent` were generalised for the `global_cardinality` constraint. The algorithm in [301] identifies `Hall intervals` and the one in [206] exploits convexity to achieve a fast implementation of the flow-based `arc-consistency` algorithm. The later algorithm can also compute `bound-consistency` for the count variables [207, 204]. An improved algorithm for achieving `arc-consistency` is described in [300].

Systems

`globalCardinality` in **Choco**, `count` in **Gecode**, `gcc` in **JaCoP**, `global_cardinality` in **SICStus**.

See also

common keyword: `count`, `max_nvalue`, `min_nvalue` (*value constraint, counting constraint*), `nvalue` (*counting constraint*),

`open_global_cardinality_low_up` (*assignment, counting constraint*).

cost variant: `global_cardinality_with_costs` (*cost associated with each variable, value pair*).

implied by: `global_cardinality_with_costs` (*forget about cost*), `same_and_global_cardinality` (*conjoin same and global_cardinality*).

part of system of constraints: among.

related: `roots`, `sliding_card_skip0` (*counting constraint of a set of values on maximal sequences*).

shift of concept: `global_cardinality_no_loop` (*assignment of a variable to its position is ignored*), `ordered_global_cardinality` (*restrictions are done on nested sets of values, all starting from first value*), `symmetric_cardinality`, `symmetric_gcc`.

soft variant: `open_global_cardinality` (*a set variable defines the set of variables that are actually considered*).

specialisation: `alldifferent` (*each value should occur at most once*), `cardinality_atleast`, `cardinality_atmost` (*individual count variable for each value replaced by single count variable*),

cardinality_atmost_partition (*individual count variable for each value replaced by single count variable and variable \in partition replaced by variable*), **global_cardinality_low_up** (*variable replaced by fixed interval*).

system of constraints: **colored_matrix** (*one global_cardinality constraint for each row and each column of a matrix of variables*).

uses in its reformulation: **tree_range**, **tree_resource**.

Keywords

application area: assignment.

characteristic of a constraint: core, automaton, automaton with array of counters.

complexity: 3-SAT.

constraint type: value constraint, counting constraint, system of constraints.

filtering: Hall interval, bound-consistency, flow, duplicated variables, DFS-bottleneck.

modelling exercises: magic series.

puzzles: magic series, autoref.

	For all items of VALUES:
Arc input(s)	VARIABLES
Arc generator	<i>SELF</i> \mapsto collection(variables)
Arc arity	1
Arc constraint(s)	variables.var = VALUES.val
Graph property(ies)	<u>NVERTEX</u> = VALUES.noccurrence

Graph model

Since we want to express one unary constraint for each value we use the “For all items of VALUES” iterator. Part (A) of Figure 5.269 shows the initial graphs associated with each value 3, 5 and 6 of the VALUES collection of the **Example** slot. Part (B) of Figure 5.269 shows the two corresponding final graphs respectively associated with values 3 and 6 that are both assigned to the variables of the VARIABLES collection (since value 5 is not assigned to any variable of the VARIABLES collection the final graph associated with value 5 is empty). Since we use the NVERTEX graph property, the vertices of the final graphs are stressed in bold.



Figure 5.269: Initial and final graph of the `global_cardinality` constraint

Automaton

Figure 5.270 depicts the automaton associated with the `global_cardinality` constraint. To each item of the collection `VARIABLES` corresponds a signature variable S_i that is equal to 0. To each item of the collection `VALUES` corresponds a signature variable $S_{i+|VARIABLES|}$ that is equal to 1.

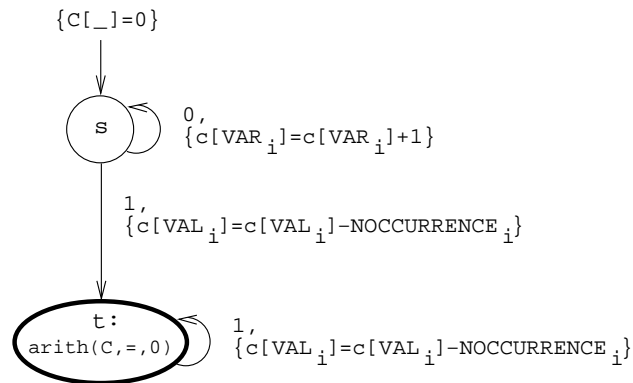


Figure 5.270: Automaton of the `global_cardinality` constraint